doi: 10.11887/j.cn.201501009　　http://journal.nudt.edu.cn

无缝坏块处理与流水编程的 NAND 型内存控制器设计与实现*

闫梦婷[1,2]，安军社[2]
（1. 中国科学院 空间科学与应用研究中心，北京　100190；2. 中国科学院大学，北京　100190）

摘　要：为满足航天大容量存储系统对高速存储及数据完整的需求，实现了一个基于 NAND 型内存的高性能控制器，提出了一种实现于 NAND 型内存芯片内部的流水编程机制，以及一种可以保证数据无缝连接的坏块处理机制。介绍了存储控制器的各个模块设计，并分析了不同情况编程机制所需的时间计算方法，建立仿真模型，利用蒙特卡洛方法仿真并讨论了流水编程机制的性能优化效果。在实际硬件平台验证了流水编程机制和坏块处理机制，结果表明该大容量存储系统的存储速率可达 100MB/s，读取数据与存入数据保持一致，数据无乱序无丢失。

关键词：NAND 型内存控制器；固态存储器；坏块处理机制；流水存储机制
中图分类号：TP333　　**文献标志码：** A　　**文章编号：**1001－2486(2015)01－053－06

Design and implement of a NAND Flash controller with pipelining program and non-missing invalid block handle method

YAN Mengting[1,2], *AN Junshe*[2]
(1. Center for Space Science and Applied Research, Chinese Academy of Science, Beijing 100190, China;
2. University of Chinese Academy Sciences, Beijing 100190, China)

Abstract: Aiming at requirement of high speed and complete of data in space storage system, the design of a high performance NAND Flash controller is present. It concludes a pipelining-programming inside of NAND Flash chip and a non-missing invalid block method. The storage implementation is present. The calculation of storage time in different situation is discussed. The simulation modules are present and the impact of pipelining programming is simulated and discussed using Monte Carlo method. Practical application proves the pipelining programming and non-missing invalid block method. The operation frequency of storage system achieves to 100MB/s, ensuring accuracy, completeness and continuity of data.

Key words: NAND Flash controller; solid storage recorder; invalid block method; pipelining program method

近年来，随着航天领域对大容量存储器研制工作的不断发展，与非型闪速存储器(no-and flash，NAND Flash)作为一种新兴的半导体存储介质，以其高密度、大容量、高数据存储速率以及可多次擦除等特点得到了大量的使用，逐步成为航天电子系统存储介质的主流选择[1-2]。文献[3]基于 NAND Flash 设计了一种高速海量存储模块，为提高存储带宽引入了并行技术，同时在时间和空间上对存储架构进行了优化；文献[4]设计了一种高速多通道并行存储架构，在文献[3]的基础上对闪存转换层算法进行了优化，最大存储速度达到 73MB/s。高速系统的设计[3-8]基本通过扩展位宽或使用多片 Flash 芯片进行流水线操作来提高存储系统的存储速率，但 Flash 芯片内部流水操作的设计尚未提出，并且存储速率会因系统的时钟频率的不同而改变。本设计利用了 NAND Flash 多片叠装结构、复用自主固化时间，实现了 Flash 芯片内流水编程机制的设计，最大存储速度达到 100MB/s，并分析了编程时间的理论值，对流水编程机制的性能优化程度进行了建模仿真，分析了流水编程机制在不同时钟频率下对系统性能的影响。

由于其制造工艺和航天器的特殊使用环境等因素，NAND Flash 可能在任意地址存在坏块，并在使用过程中产生新的坏块[9-10]，对坏块的处理是 NAND Flash 实际应用中必须要解决的问题。大量的 Flash 使用实例[3-8]中已出现常见的坏块处理方法，主要是在使用前建立坏块信息表，以此

---

* 收稿日期：2014－06－30
基金项目：中国科学院战略性先导科技资助项目(XDA04060300)
作者简介：闫梦婷(1989—)，女，广西桂林人，博士研究生，E-mail: mengtingyan@outlook.com;
安军社(通信作者)，男，研究员，博士，博士生导师，E-mail: anjunshe@nssc.ac.cn

来检索有效块，屏蔽坏块。根据坏块信息表建立方式可以分为两类[8]：1) 直接存储，即只存储有效块的地址，剔除无效块地址; 2) 间接映射，即在块信息表空间相应的地址进行标记，以此来判断对应块的性质。根据坏块存储区域可以分为: 控制器内部存储以及 Flash 页内存储，后者可以节约控制器资源，但访问块信息表的时间过长，不适合于高速系统。另外，还有基于文件分配表(File Allocation Table，FAT) 文件系统处理坏块方法，软件操作却存在一定难度[9]。上述已实现的坏块处理方法能屏蔽已知坏块，但在新坏块产生后会丢弃整个坏块，并未涉及对坏块内已存数据的恢复处理，一旦在编程操作中出现坏块就会丢失数据，最坏情况是丢失一个块容量的数据。本文设计了一种数据无缝连接的坏块处理机制，在使用过程中产生新的坏块时能够使数据恢复，保证数据完整性。结合流水编程机制和坏块处理机制，本设计实现的大容量存储系统提高了整体性能，满足实际航天电子系统对高速存储、数据完整无丢失的要求[11－13]。

1　NAND Flash 控制器整体设计

本设计中的 NAND Flash 控制器内部分为初始化、擦除、编程、读取、标记以及主控制器六个模块。如图 1 所示，系统上电后自动进入初始化模块，初始化模块的任务是读取所有的 Flash 块的坏块标记字节以及 Spare 区域的用户使用信息，生成形成完备的块信息表，后续存储系统的全部工作都将根据块信息表来执行。本设计为达到存储系统对高速率的要求，必须减少读取块信息表所需的时钟周期，因此选择在控制器内部建立 RAM 区用来存放坏块信息表。对 Flash 的基本操作有编程、读取和擦除，三种操作各自为一个模块。另外，标记模块就是根据擦除模块和编程模块提供的坏块地址对坏块进行标记，主控制器模块则控制协调所有子模块的工作。

编程和读取操作的基本单位为页，每次编程读取操作存储、读取一页容量的数据以及使用信息。编程操作模块中，为了提高存储系统的整体编程速率，利用了流水操作的方法在片内实现了流水编程操作，并且嵌入了完备的坏块处理机制用于处理编程操作中产生坏块的数据丢失问题，编程模块将在下文详述。读取模块会在接收到读取指令后，按照顺序从 Flash 相应页地址中将数据读出发送给接收器件。Flash 存储单元在执行数据写入之前需要进行擦除操作，擦除模块完成

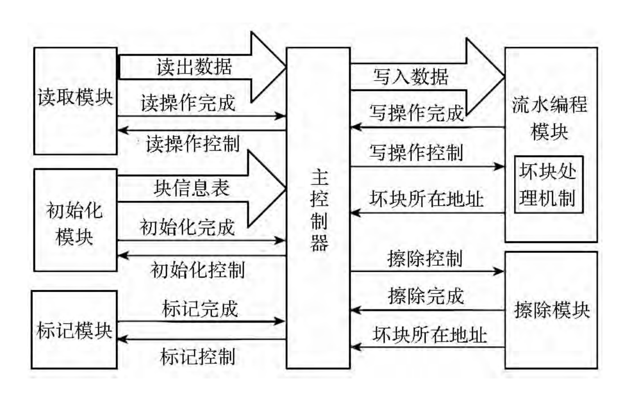


图 1　控制器模块组成示意图

Fig. 1　Modules of controller

以块为单位的擦除操作，擦除操作可能导致新的坏块产生，此时需要启用标记模块对坏块进行标记。擦除操作实际上是将把整块 Flash 的全部字节单元恢复为“FF”，即把所有的位全部恢复为“1”。编程操作则是将字节单元中相应的位从“1”置为“0”，以此将每个字节编程为相应的十六进制数[10]。而坏块的常见情况是无法将字节单元里的“1”值置为“0”，或者无法将“0”的位恢复为“1”，对坏块进行编程操作，将导致存入数据与实际编程成功的数据不一致，读出错误数据，故而对 Flash 的使用要避免使用坏块。

2　流水编程机制的设计

2.1　NAND Flash 叠装芯片的结构特点

NAND Flash 芯片内部架构为多片 Flash 晶片叠装，本设计选用 8 片叠装式的 Flash 芯片，每单片 Flash 包含 4096 个块，1 块(block) 包含 64 页(page) ，1 页包含 2048 字节数据区以及 64 字节 Spare 区，其中每页的数据区用于存放数据，Spare 区则是提供给用户存放使用信息[11－12]。Samsung Electronics 的型号为 K9K8G08U1A 的 Flash 芯片的架构图如图 2 所示。8 个单片 Flash 共用的信

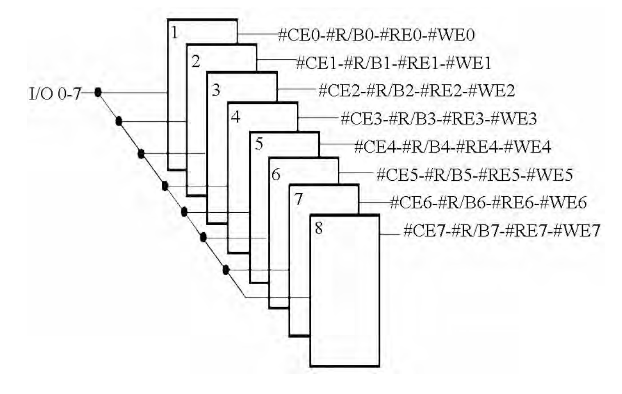


图 2　NAND Flash 8 片叠装内部示意图

Fig. 2　8 slides inside of NAND Flash

号有: 指令锁存使能信号(CLE) 、地址锁存使能信

号(ALE) 和一组 8 位的 I/O 接口。其他控制信号包括 CE、RE、WE、R/B 则是 8 片 Flash 各有一组。片选信号 CE 用于激活 8 片 Flash 晶片中的某一片，使得共用信号与其连接，流水操作就是利用片选信号依次激活内部分立叠装的 Flash 晶片，将自主固化时间得以复用来提高存储系统的整体编程效率。

2.2　流水编程机制设计

根据 Flash 的编程操作时序，控制某一片 Flash 某一页进行编程时需要对 Flash 芯片的操作如下: 1) 发送起始指令和地址; 2) 加载 2048 字节数据和 Spare 区的若干字节使用信息; 3) 发送结束指令[9]。每一片 Flash 芯片内部都有一个数据缓存区，可以用于缓存一页最大容量的数据，控制器向 Flash 加载的数据实际上是暂存在数据缓存区里，当接收到结束指令后，Flash 进入自主固化阶段，将数据缓存区的数据按顺序固化到当前物理地址指向的页空间中。在 Flash 芯片的自主固化时间内，控制器无须对该片 Flash 进行控制操作，连接该片 Flash 的所有信号处于闲置状态。根据 Flash 厂家提供的数据显示，Flash 固化时间的典型值约为 300 μs，最大值为 700 μs[9]。如果不采用流水编程机制，对 Flash 的所有页空间进行普通编程，写入每一页数据都需要等待 Flash 自主固化完成，使得存储系统的整体编程效率低，无法实现高速存储的目的。因此本设计引入流水操作概念，在多片叠装式 NAND Flash 芯片内实现了流水编程机制，大幅度地提高存储系统的整体编程速率[13]。

本设计对 8 片叠装的 Flash 芯片采取流水方式进行编程操作，每次对 8 片 Flash 的同一逻辑块地址的同一页进行编程操作，并将此称为一个流水级。CE0 ~ CE7 为 8 片 Flash 对应的片选使能信号，为了叙述简便，将 8 片 Flash 单片分别定义为 CE0 ~ CE7。每个流水级的控制器操作包括两部分。1) 控制编程操作部分: 依次对 CE0 到 CE7 发送起始指令并加载数据，使得 8 片 Flash 芯片依次进入自主固化阶段。2) 编程完成情况检测部分: 编程操作完成之后需要对编程完成的状态进行检测，依次对 CE0 到 CE7 已完成编程操作的芯片发送检测指令，根据芯片返回状态判断是否产生新的坏块。如果检测某一片 Flash 时该片仍处于自主固化阶段，则需要等待其固化完成后再进行状态检测。流水编程工作顺序如图 3 所示。当 CE0 ~ CE7 全部检测到编程成功，则一个流水级的编程工作完成[14]。

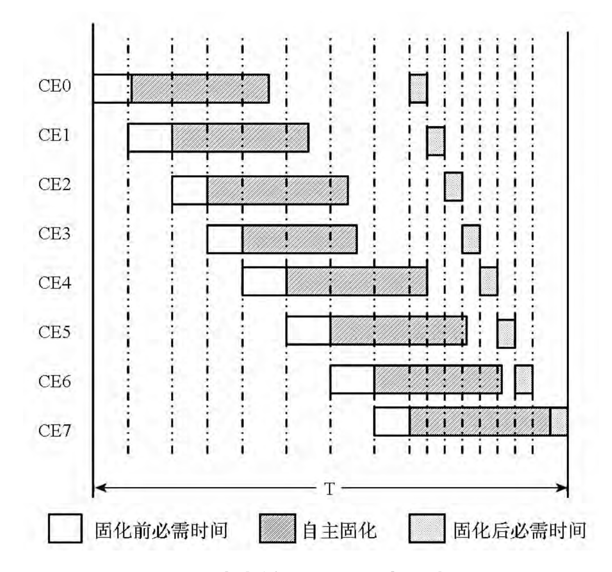


图 3　流水编程工作顺序示意图

Fig. 3　Pipelining-programming method

2.3　系统编程所用时间的计算方法

为了从理论上验证流水编程机制对存储系统性能的改善，分析了不同情况下编程时间的计算方法，建立了模型对此进行了仿真，并且讨论了存储系统时钟频率的不同、自主固化时间的不稳定，对流水编程机制的性能影响。

设固化前发送指令和数据的时间为 a，固化后发送检查指令和状态返回时间为 b，自主固化时间为 X，则第 i 片 Flash 固化时间可以表示为 X_i。

若不采用流水编程机制，则 8 次页编程完成的时间 T_1 可以表示为:

$$T_1 = 8 \times a + \sum_{i=0}^{7} X_i + 8 \times b \tag{1}$$

其中，Xi 表示第 i 片 Flash 的固化时间，$i = 0, 1, \cdots, 7$; a, b 分别表示固化前控制必需时间和固化后控制必需时间。

若采用流水编程机制，则 8 片 Flash 页编程完成的时间 T_2 可以表示为:

$$T_2 = \begin{cases} 8 \times a + 8 \times b, & \forall X_i < Y_i \\ (j+1) \times a + X_j + (8-j) \times b, & \forall X_i \geqslant Y_i \end{cases} \tag{2}$$

其中，$Y_i = (7 - i) \times a + (i + a) \times b$，$i = 0, 1, \cdots, 7$; 令 $Z_j = X_j + (j + 1) \times a$，$X_j$ 为 Z_j 取得最大值时对应的 X 值。

当 $X_i < Y_i$ 对于所有的 X_i 都成立时，很明显看出 T_2 比 T_1 节省了所有的固化时间。当 $X_i < Y_i$ 不能全部 X_i 成立时，式(2) 得出 T_2 的取值由 X_j 以及 X_j 处于流水级中的位置联合决定，为准确直观地表现流水编程机制的性能优化程度，根据上述

理论分析建立了仿真模型，采用不同时钟频率以及不同自主固化时间进行了仿真测试，结果将在下文中进行讨论。

3 无缝连接式坏块替代机制

3.1 坏块产生对编程的影响

Flash 在使用过程中会随机产生新的坏块，编程操作和擦除操作都会导致坏块产生[11-12]，其中擦除操作产生的坏块只需要对该坏块进行标记并及时更新块信息表将其屏蔽，但是编程操作产生的坏块涉及到数据的完整性以及数据读取的连续性，需要提出一个处理机制来应对编程操作产生的坏块。固化完成后 Flash 接收状态读取指令，若状态显示为编程失败，表明当前页的数据没有正常写入，称该页为编程失败页。编程失败页中的数据为无效数据，不能作为有效数据直接读取，加载到该页的数据将丢失。编程失败页所在的块为编程失败块，即新出现的坏块，若直接将编程失败块屏蔽不再使用将导致已正常编程页的数据全部丢失，最坏情况会丢失整个块容量的数据，将导致整个存储系统的数据丢失情况严重加剧。为了保证数据不丢失，本设计提出了一个无缝连接的坏块处理机制，流水编程机制，能够在高速编程过程中实现完备的坏块处理。

3.2 坏块处理机制的设计

在本设计中，如果 8 片 Flash 的页编程操作都成功，则称这一级流水编程成功; 若有某一个 Flash 晶片的页编程操作失败则称该流水级编程失败，开始调用坏块替代机制。流水编程的替代机制具体流程如图 4 所示。

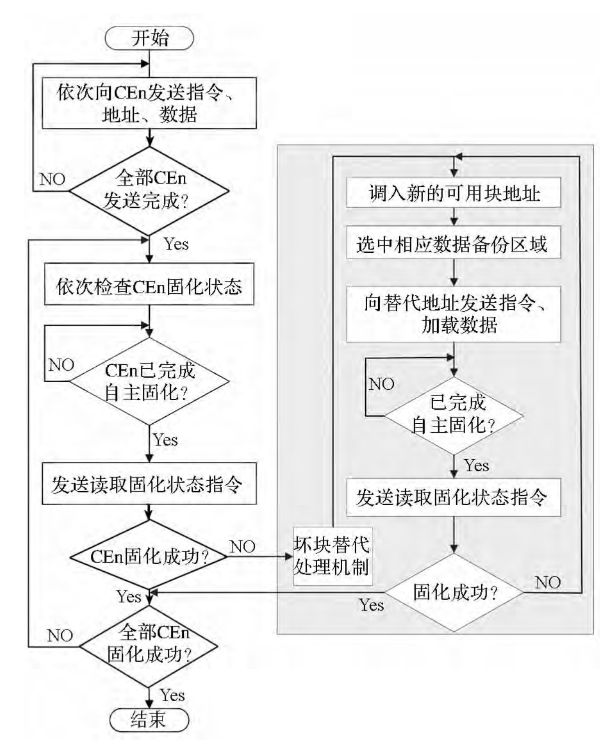


图 4 流水编程替代机制流程图

Fig. 4 Invalid block in pipelining programming method

无缝连接坏块处理机制的具体设计涉及到数据备份和 Flash 编程操作地址。为了保证不丢失编程失败页中的数据，需要进行数据备份，在控制器内部例化出 RAM 空间分区后用于数据备份。当执行数据加载工作时，同时将数据写入到对应的 RAM 区里。发生页编程失败后，调用相应的 RAM 区里的数据再次写入到替代块的替代页中。只有在确保 RAM 区里的数据编程成功之后才能允许数据被覆盖。

Flash 编程的地址指向即将执行编程操作的页单元，页地址在每次编程完成后加一指向下一页; 当 64 页全部写满时提取新的块地址，块地址需要检索初始化生成的块信息表得到。本设计提出逻辑块地址概念，对存有数据的物理块标记上逻辑块地址，作为有效块的顺序。发生页编程失败的坏块和该坏块的替代块拥有同一个逻辑块地址，替代块的第一个编程的页地址为坏块中编程失败的页地址。使得坏块与替代块无缝连接后，相当于一个有效块，在后续的数据读取工作中，根据唯一的逻辑块地址可以顺序回放出存储的数据。

3.3 数据的读取顺序处理

数据的读取顺序将结合流水编程机制索引逻辑块地址得到。将每一页的数据看成一个包，设第 n 包存放在第 A 片，第 $B(b)$ 块的第 C 页，B 表示物理块地址，b 为逻辑块地址。第 n 包数据的存放地址可表示为:

$$n = (A+1) + 8\times 64\times (B+1) + 8\times (C+1) \tag{3}$$

如果不产生坏块，则 $B=b$，物理块地址与逻辑块地址相同。一旦编程操作中产生了新的坏块，检索出可用的块地址作为替代块，对替代块标记当前使用的逻辑块地址。假设第 n 包数据发生页编程失败，则第 n 包数据第一次编程存放的地址表达式为:

$$n = (A+1) + 8\times 64\times (B_i+1) + 8\times (C+1) \tag{4}$$

检测到该地址编程失败，调用替代块进行再次编程的地址表达式为:

$$n = (A+1) + 8\times 64\times (B_{i+1}+1) + 8\times (C+1) \tag{5}$$

其中 B_{i+1} 和 B_i 的逻辑块地址都为 b_i。

如图 5 所示，物理块地址 B 之前都未发生页编程失败，直到物理块地址 B 的 CE3 单片 Flash 的第($n-3$) /8 页发生了页编程失败，于是将第 n 包数据再次编程到替代块中，可见替代块的物理块地址为 $B+1$(假设该块为有效块) ，逻辑块地址仍为 b，并且后续的编程操作将用替代块代替坏块继续进行，直到该逻辑块全部写满。图 5 中由 B 与 $B+1$ 物理块组合而成的逻辑块 b，无缝连接后实际上相当于一个正常块，根据逻辑块地址寻址读数，可以保证 B 物理块中的已存数据不丢失，也能保证流水编程操作顺序执行。

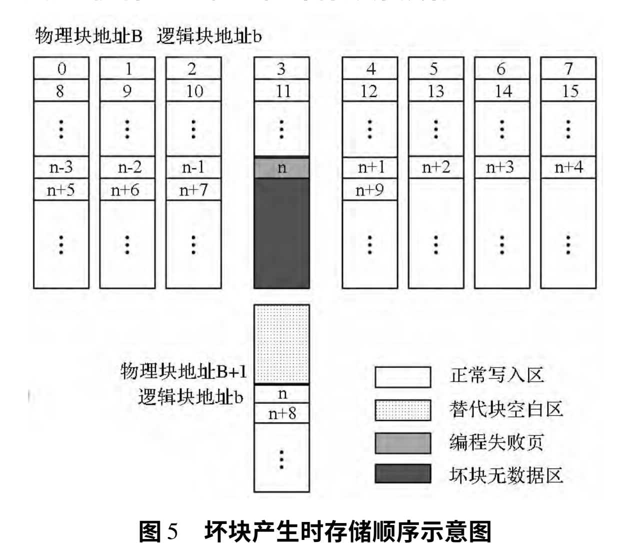


图 5　坏块产生时存储顺序示意图

Fig. 5　Storing order when invalid block occurs

4　测试结果与分析

4.1　时钟频率对流水编程机制性能的影响

选取 4MHz ~ 64MHz 范围内的常用时钟频率，并根据向 Flash 发送指令、地址以及加载数据所需周期计算出不同时钟频率下 a，b 的取值。

Samsung Electronics 给出的 NAND Flash 芯片测试出的典型自主固化时间为 300 ±100 μs，超长自主固化时间最大值为 700 μs [9]。首先讨论自主固化时间 X_i 不出现超长固化时间的情况下流水编程机制的编程速率。建立仿真模型并采用蒙特卡洛方法，该方法利用重复多次仿真数据的方差来代表被测对象的平均水平，令 X_i 在典型值[200，400] μs 之间取随机值，多次测试结果如图 6 所示。从图中可以看出，在相同频率系统时钟下，采用流水编程机制的系统比普通编程机制所需的编程时间明显减少；并且随着系统时钟频率的提高，普通编程机制所需时间趋于平衡，但流水编程机制所需时间的性能仍有较大优化效果。

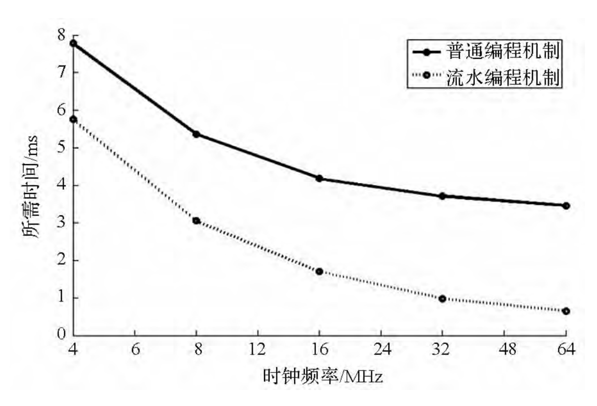


图 6　典型固化时间对比图

Fig. 6　Typical storage time comparison

下面讨论编程中出现超长自主固化时间的情况。如图 7 所示，在普通编程机制中，出现超长自主固化时间将直接影响整体编程速率，系统所用的编程时间将随着出现超长固化时间的页编程个数线性增长。采用了流水编程机制后，根据式(2) 和式(4) ，在一个流水级中，所用的编程时间与出现的超长固化时间的页编程个数无关，只与式(4) 取得最大值的超长固化时间有关。得出结论，在一个流水级中，当出现一个或多个超长固化时间，流水编程机制比普通编程机制所需编程时间明显减少。

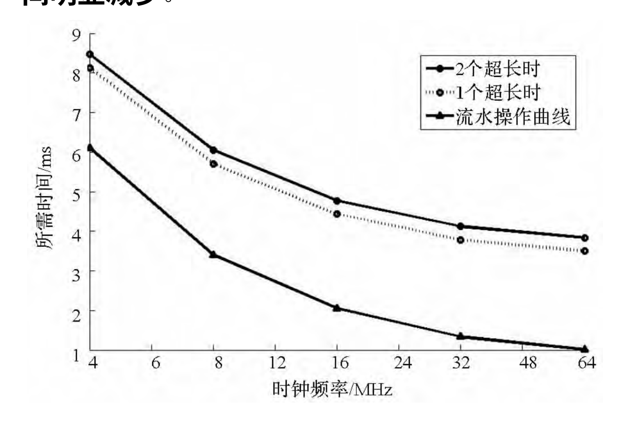


图 7　超长固化时间对比图

Fig. 7　Overtime storage time comparison

4.2　坏块替代处理机制测试与仿真结果

本设计选用了 Xilinx 公司型号为 Virtex 5 - xc5vlx85FPGA 硬件平台，将 Flash 控制器模块下载到可编程器件后所占用逻辑资源为 7.5% [15]。进行连接 CPU、数据源和接收器等单元配合的板级测试，利用数据源进行大容量数据存储测试，遍历所有 Flash 存储单元、多次测试结果见表 1：在频率为 16MHz 的系统时钟下，验证本设计的存储系统可以实现的平均存储速率可达 100MB/s 以上。存取数据对比结果显示，本设计的大容量存储器能够检测出已知坏块，并屏蔽已知坏块，并在

新坏块产生时调用坏块处理机制，保证无数据丢失，实现数据的无缝连接。

表1　平台测试统计数据

Tab.1　Data statistics in test bench

遍历次数	坏块个数	平均存储速率(MB/s)
1~10	43	125
11~20	135	112
21~30	201	116
31~40	263	131

5　结论

流水编程机制使得存储系统的整体编程速率大幅度提高，并且时钟频率越高，性能优化程度越高；在出现较长固化的特殊情况时，仍然可以保证整体编程速度优于普通编程机制。提出了无缝连接式坏块处理机制，能够提供一个完备的方案实现突发坏块时对数据的无缝处理，改善了常用坏块处理机制无法恢复坏块数据的情况；并利用逻辑块地址索引进行读取操作，能够保证无丢失无乱序地存储读出数据，使得整个存储系统性能完好。

参考文献(References)

[1] 王立峰，胡善清．基于闪存的高速海量存储模块设计[J]．计算机工程，2011，37(7)：255－257.
WANG Lifeng, HU Shanqing. Design of high speed and large capacity storage module based on flash memory[J]. Computer Engineering, 2011, 37(7)：255－257. (in Chinese)

[2] 彭军，黎福海．一种多通道并行固态存储系统的设计与实现[J]．计算机工程，2013，39(12)：40－44.
PENG Jun, LI Fuhai. Design and implementation of a multi-channel parallel solid state storage system[J]. Computer Engineering, 2013, 39(12)：40－44. (in Chinese)

[3] 张胜勇，高世杰．基于FPGA的NAND Flash坏块处理方法[J]．计算机工程，2010，36(6)：239－243.
ZHANG Shengyong, GAO Shijie. Bad block handle method of NAND Flash memory based on FPGA[J]. Computer Engineering, 2010, 36(6)：239－243. (in Chinese)

[4] 贾源泉，肖侬．基于NAND FLASH的多路并行存储系统中坏块策略的研究[J]．计算机研究与发展，2012，49(9)：68－72.
JIA Yuanquan, XIAO Nong. Research on bad block of the parallel storage system based on NAND FLASH[J] Journal of Computer Research and Development, 2012, 49(9)：68－72. (in Chinese)

[5] Samsung Electronics. 512M × 8 Bit/1G × 8 Bit NAND flash memory(Revision 0.2)[R]. 2003.

[6] Takeuchi K. Novel co-design of NAND flash memory and high-speed solid-state drives[J]. VLSI circuit, 2009, 44(4)：1224－1227.

[7] NAND Flash Applications Design Guide[EB/OL]. (2004－05－19)[2013－10－21]. http://pdf1.alldatasheet.com/datasheet-pdf/view/85039/SAMSUNG/K9W4G08U1M.html.

[8] Shibata N, Maejima H, Isobe K, et al. 70nm 16Gb 16-level-cell NAND Flash memory[J]. IEEE Journal of Solid-State Circuits, 2008, 43(4)：929－937.

[9] Takeuchi K. Novel co-design of NAND flash memory and high-speed solid-state drives[J]. VLSI circuit, 2009, 44(4)：1224－1227.

[10] 邢开宇，曹晓曼，方火能．基于FPGA和NAND FLASH的存储器ECC设计与实现[J]．电子科技，2012，25(10)：69－73.
XING Kaiyu, CAO Xiaoman, FANG Huoneng. ECC design based on FPGA and NAND flash memory[J]. Electronic Science and Technology, 2012, 25(10)：69－73. (in Chinese)

[11] Virtex-5 FPGA configuration user guide. [EB/OL]. (2012－10－19)[2013－10－21]. http://www.xilinx.com/support/documentation/user_guides/ug191.pdf.

[12] 潘立阳，朱钧．FLASH存储器技术与发展[J]．微电子学，2002，32(1)：1－6.
PAN Liyang, ZHU Jun. FLASH memory technology and its development[J] Microelectronics, 2002, 32(1)：1－6. (in Chinese)

[13] 梁永刚，崔永俊，郇弢．基于NAND型FLASH的双备份固态存储系统[J]．科学技术与工程，2013，13(26)：7676－7681.
LIANG Yonggang, CUI Yongjun, HUAN Tao. Double backup solid-state storage system design based on the NAND-type flash[J]. Science Technology and Engineering, 2013, 13(26)：7676－7681(in Chinese)

[14] Nguyen Q, Yuknis W, Pursley S, et al. A high performance command and data handling system for NASA's lunar reconnaissance orbiter[C]//Proceedings of AIAA Space 2008 Conference & Exposition, Washington, 2008.

[15] Durna M, Urhan H, Turhan O, et al. A new generation on-board computer and solid state data recorder suitable for spacewire platforms[C]//Proceedings of 3rd International Conference on Recent Advances in Space Technologis, RAST, 2007.